REINFORCED SAMPLE REWEIGHTING POLICY FOR SEMI-SUPERVISED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing semi-supervised learning (SSL) has been shown to be an effective paradigm for learning with less labeled data. To improve the performance of SSL, existing methods build sample reweighting or thresholding strategies to handle the category bias or erroneous pseudo labels. However, most of these existing methods are based on the heuristic hand-crafted rules, which require laborious adjustment, and may lead to sub-optimal solutions that cannot improve the model performance to the greatest extent. Here, to the best of our knowledge, we pioneer to develop an automatic strategy that boosts the performance of SSL. We introduce an end-to-end sample reweighting policy for semi-supervised learning, with a delicately designed Markov Decision Process (MDP) framework. The MDP framework is constructed with an agent network, which is optimized in a reward-driven manner, and receives the carefully designed state and action representations for decision reference. We also design a memory paradigm for computation-efficient representation construction and MDP solving. We further introduce a "pretraining-boosting" two-stage MDP curriculum where the agent network is firstly pretrained and then optimized continuously in the deployment phase to catch up with the constantly updated classification network. Extensive experiments demonstrate that our method achieves state-of-the-art performance on multiple datasets, outperforming previous advanced approaches such as FixMatch.

1 INTRODUCTION

Although having achieved great success, deep learning, in many cases, is plagued by the need for a huge amount of labeled data. To this end, semi-supervised learning Sohn et al. (2020); Zhang et al. (2021a); Xie et al. (2020); Gong et al. (2021); Berthelot et al. (2019b;a) has attracted much attention in recent years due to its ability to leverage not only the labeled but also the unlabeled data for improving model performance.

Currently, most of the semi-supervised learning methods Sohn et al. (2020); Zhang et al. (2021a); Xie et al. (2020) are based on the "pseudo label training" mechanism with a teacher-student network structure, in which the teacher network generates pseudo labels, and the student network uses them for training. However, such a scheme faces challenges including erroneous pseudo labels because of wrong predictions from the teacher network, and the category bias due to the class imbalance of the training data. To address these issues, Sohn et al. (2020); Xie et al. (2020); Berthelot et al. (2019a) build hard or soft thresholds and weighting scores to emphasize samples with cleaner labels based on the entropy confidence. Meanwhile, Zhang et al. (2021a); Hu et al. (2021) set different thresholding strategies for different categories to achieve category-unbiased training. Despite achieving significant improvement, these existing methods are generally based on the hand-crafted rules, which are often manually designed based on the heuristic inference. However, such hand-crafted rules that often need careful adjustment may not lead to optimal solutions for improving performance to the greatest extent. For example, it is hard to find the optimal thresholds that can keep the trade-off between the cleaner labels and the sufficient training samples.

In this work, to address the above-mentioned issues, we free the need for hand-crafted strategy designing such as the manually set thresholds and weighting functions. Instead, we propose a novel method by performing sample reweighting for semi-supervised learning with an agent network, to automatically emphasize different samples with different degrees based on their potential contributions.

The agent network learns a sample reweighting policy in a reward-driven manner by maximizing the performance on an individual set, so that the obtained policy is directly performance improvement-related. With the learned policy, the agent generates a reweighting score for the training of each unlabeled sample, without the need for the manually designed thresholds or weights adjustment strategies. Such a learning procedure can be formulated as a Markov Decision Process (MDP), which is solved with Reinforcement Learning (RL). The agent receives state and action representations for generating weighting scores. To ease the weighting decision of the agent, we carefully design the state and action, so that they contain essential information such as pseudo label confidence and class-wise accuracy. These information can reflect the contribution and importance of each sample so that they can guide the agent for making a better decision. To relieve the computation burden of constructing state and action by calculating on the whole dataset, a queued memory is also designed in our framework. We further propose an effective dual-phase MDP method, which consists of not only a common training phase for pretraining an initialized policy, but also a deployment boosting phase where the agent is continuously boosted with the updating of the classification network. Such a dual-phase mechanism enables boosted performance benefited from the dynamic adaptation to the training status.

To the best of our knowledge, our work pioneers to deploy an automatic end-to-end policy with the task-specific state and action representations for semi-supervised learning. Benefited from the reward-driven optimization, the automatically learned policy results in better performance compared to the hand-crafted counterparts. The learning policy can also be dynamically adapted and optimized going along with the training process via our "pretraining-boosting" two-stage MDP curriculum. Experimental results have demonstrated the effectiveness of our designed framework. Our method achieves state-of-the-art performance on multiple datasets, outperforming previous advanced approaches such as FixMatch Sohn et al. (2020).

In summary, our main contributions are: **1)** We propose an MDP framework for semi-supervised learning. This framework can automatically generate a weighting score for each sample via an agent network, without the need for the hand-crafted adjustment strategies. This framework is achieved in a reward-driven manner, so that the obtained policy is directly optimized to be performance improvement-related. **2)** The MDP framework is delicately designed in the network structure with effective reference information to guide weighting decisions, the training strategy with queued memory method to reduce the computation cost for constructing state and action representations, and the deployment stage with a "pretraining-boosting" curriculum. **3)** We evaluate our method on multiple datasets and achieve state-of-the-art (SOTA) performance.

2 METHOD

2.1 PRELIMINARIES

We follow the settings of Sohn et al. (2020); Zhang et al. (2021a); Xie et al. (2020) to construct the basic framework for semi-supervised learning as follows. Given the labeled dataset $D_l = \{(x_i^l, y_i^l)\}$ and the unlabeled dataset $D_u = \{(x_i^u)\}$, the semi-supervised learning network processes the two alternately, and can be optimized in a teacher-student learning manner. Specifically, for the labeled data, a common cross-entropy loss is calculated as the supervised loss $\mathcal{L}_l$. For the unlabeled data, the teacher network g is input with the weak augmentation of the unlabeled image x_i^u and generates the corresponding pseudo label $\hat{y}_i^u$, while the student network f is input with the strong augmentation of x_i^u and uses $\hat{y}_i^u$ as the supervision for network training. Then the unsupervised loss for each mini batch is formulated as:

$$\mathcal{L}_u = \frac{1}{N_b} \sum_{i=1}^{N_b} L_u^i, \quad L_u^i = L_{ce} \left(f\left(x_i^u\right), \hat{y}_i^u \right), \tag{1}$$

where L_{ce} denotes the cross-entropy loss and N_b denotes the number of images in each batch. The final loss for training the student network equals to $\mathcal{L}_l + \mathcal{L}_u$. And the parameters of the teacher network are updated by the exponential moving average of the student network. Finally, the student network is used for inference. Such a basic framework faces challenges including the noisy pseudo labels and the class bias issues, resulting in unsatisfactory performance, as reported by Xie et al. (2020); Zhang et al. (2021a); Hu et al. (2021). To address this issue, some recent works Zhang et al. (2021a); Hu et al. (2021); Gong et al. (2021) focus on performing sample reweighting for the

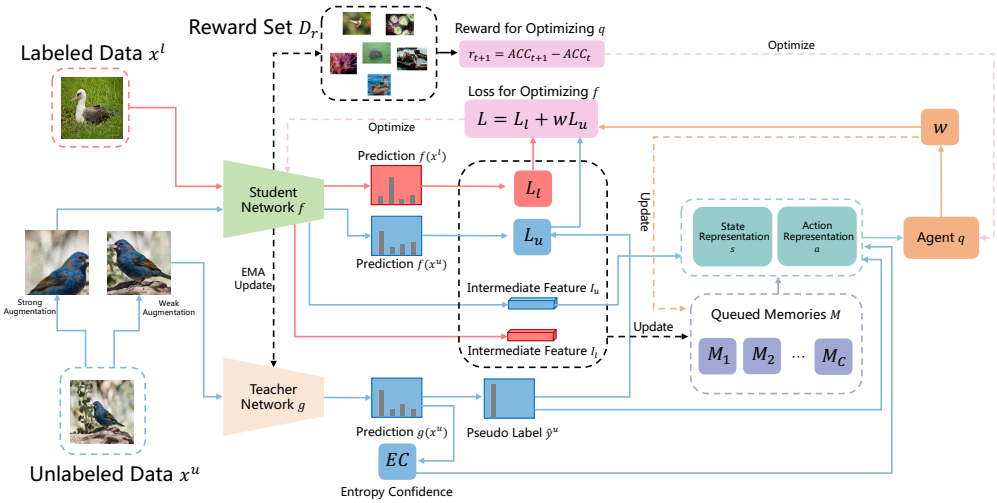


Figure 1: Overview of our proposed method for semi-supervised learning. To simplify the illustration, this figure is based on the setting where one training batch has one labeled data x^l and one unlabeled data x^u, which are sampled from the labeled training set D_l and the unlabeled training set D_u, respectively. x^l passes through the student network f and gets the supervised loss L_l. The weak augmentation of x^u is fed into the teacher network g to get the pseudo label $\hat{y}^u$, which is used as the supervision of the prediction results of f that takes the strong augmentation of x^u as the input. The intermediate feature I_u, pseudo label $\hat{y}^u$, entropy confidence EC along with the queued memories are employed to compute the state and action representations, which are then fed into the agent q, getting the weighting score w. The weighted loss denoted as $L_l + wL_u$ is used to optimize f. The reward r_{t+1} by measuring the accuracy difference after one updating step of f is used to optimize q. The parameters of teacher network g is updated by the exponential moving average (EMA) of f. The queued memories are updated by L_l, L_u, I_l, I_u and w representing the current status.

unlabeled samples, with the following modified loss with the weighting score w_i introduced:

$$\mathcal{L}_u = \frac{1}{N_b} \sum_{i=1}^{N_b} w_i L_u^i, \quad L_u^i = L_{ce}\left(f\left(x_i^u\right), \hat{y}_i^u\right), \tag{2}$$

where w_i is designed to filter out the noisy labels or construct a class-balanced training set. Existing works mainly use manually designed heuristic rules to generate the weighting scores, e.g, through a threshold based on pseudo labels' confidences to form a binary weight Xie et al. (2020), or a class-dependent weight value according to each class's training status Hu et al. (2021). However, these heuristic rules are hand-crafted and require careful empirical analysis. Because of the manually designing scheme, these heuristic hand-crafted rules may not be the optimal solutions for network optimization. This issue motivates us to propose a brand new learning method – by automatically learning a reweighting policy through reinforcement learning. Therefore, in this work, an agent network q is introduced to learn the policy and generate the weighting score for each unlabeled sample. Such an optimization process is reward-driven, by maximizing the reward calculated from an individual dataset. Such a reward can reflect the testing performance improvement, so that the learned reweighting policy can directly improve the performance. The overall structure of our method is shown in Fig. 1.

2.2 Markov Decision Process for Semi-supervised Learning

To automatically learn a reweighting policy, we design a reward-driven method, in which the reward should reflect the model performance improvement. To achieve this, we separate 25% samples from the labeled set to form a reward set D_r, and define the rest of the labeled samples as the labeled training set D_l. D_l along with the unlabeled training set D_u are used to train the classification network, while D_r is used to compute the reward by evaluating the trained classification network on it.

We formulate the sample reweighting process in semi-supervised learning as a Markov Decision Process (MDP). For the t-th iteration, the corresponding MDP can be defined as a sequence $(s_t, a_t, r_{t+1}, s_{t+1})$, where s refers to the state, a refers to the action and r denotes the reward. Both the state and action are information captured by the classification network. Then each iteration for agent training can be divided into the following five steps: (1) Computing the state s_t and action a_t of the current stage by using the input images with the classification network. (2) Using the agent q to produce the weighting score w for each unlabeled sample with the input s_t and a_t. (3) Retraining the classification network f_t using the weighted loss as in Equation. 2, which results in f_{t+1} and the updated state s_{t+1}. (4) Computing the reward r_{t+1} on the reward set D_r using the network f_{t+1} and f_t. (5) Updating the agent through the reward r_{t+1}.

We employ the Q-learning algorithm Van Hasselt et al. (2016) to solve the MDP. Specifically, we optimize the agent so that the reward is maximized based on the network trained with the learned reweighting policy. More concretely, such an optimization process is implemented by minimizing the following temporal difference (TD) error:

$$TD\left(\theta, \hat{\theta}\right) = \left(r_{t+1} + \gamma q\left(s_{t+1}, a_{t+1}; \hat{\theta}\right) - q\left(s_t, a_t; \theta\right)\right)^2, \tag{3}$$

where θ denotes the parameters of the agent's policy network, $\hat{\theta}$ refers to the parameters of the agent's off-policy parameters. $\hat{\theta}$ targets at saving the learned Q-value, and is periodically updated based on θ. We calculate the reward by evaluating the accuracy gain on D_r after an updating step of the classification network from f_t to f_{t+1}, which is introduced detailedly in Sec 2.5.

Employing such an MDP framework into semi-supervised learning is nontrivial. Specifically, we face three key challenges for designing this reward-driven method: 1) How to design the state and action so that there are sufficient reference information for the agent to generate weighting scores. 2) How to process the reference information under a reasonable computation cost. 3) How to deploy the agent into semi-supervised learning where the classification network is continuously updated. In the following part, we address the above-mentioned challenges throughout the designing of our method.

2.3 State and Action

The agent network receives the state and action representations to generate weighting scores. Therefore, the state and action not only form two components of the MDP framework, but also work as reference information to guide the weighting decision of the agent. Thus, we capture reference information that determines the contribution to the performance improvement through a specific sample, and integrate the information into the state and action representations. Specifically, the state and action are considered to reflect the *global status* of the dataset and the *sample-specific properties*, respectively. Below we introduce the design details for the state and action representations.

State. Semi-supervised learning often suffers from the problem of class bias, as reflected by the different training accuracies for different categories. Thus, training using the class independent strategies can be sub-optimal. For example, using the same confidence thresholding strategies for all classes as in Sohn et al. (2020) can lead to an imbalanced training set, since fewer samples within the hard classes will be selected due to the generally lower confidences of these classes. The negative effect caused by such an issue has been found in Zhang et al. (2021a). Motivated by this, to help get a class-specific strategy, we capture the category-level performance that reflects the properties of each class from a global view. Intuitively, samples belonging to the categories with worse performance should get higher weighting scores, to encourage the optimizer to put more attention on them. Therefore, the *class-bias information* is considered to be able to help the agent to produce a class-unbiased policy for improving performance. To extract this information, we calculate the **average accuracy** of all samples belonging to each category. The higher average accuracy reflects the better performance for a certain class. Finally, we concatenate the average accuracy of all classes, thus forming the state representation reflecting the global status of the dataset.

Action. The action representation reflects how an individual unlabeled sample along with its pseudo label contributes to the training process. **Confidence** and **diversity** play important roles in such contribution measurement. Specifically, confidence measures the possibility that a pseudo label is correct, while diversity reflects how an unlabeled sample is different from other samples belonging to the same category. Intuitively, the unlabeled samples with higher confidence and higher diversity should get higher weighting scores, since training using such samples brings less noise due to the

more reliable pseudo labels, and helps to avoid over-fitting due to the higher novelty. Therefore, for the unlabeled sample, we capture confidence and diversity information as the action representation to guide the agent. For confidence, we calculate the entropy of the prediction result from the teacher network. For diversity, we use the cosine similarity of the intermediate features (features before the fully-connected layer of the CNN) to measure the diversity of every sample. We calculate the diversity between each unlabeled sample with other samples within the same class, getting a set of similarity values. Finally, diversity information can be represented as the statistical properties of these values.

Reducing Computation via Queued Memories. To capture the reference information reflecting the current status, the state and action representations introduced above need to be calculated at each training iteration. However, such calculation brings unbearable high computation cost, since all samples need to be fed into the new network to get the updated accuracy and intermediate features. Such high computation cost is unacceptable for training, so that we propose a memory mechanism to relieve the computation burden. Specifically, for a dataset with C classes, we introduce C queued memories $M = \{M_c\}_{c=1}^{C}$ to store the information of the recent training samples for each class. Each category corresponds to one memory, storing the information of both the labeled and unlabeled samples. Therefore, we can formulate each M_c for the c-th category as $\{\{L_l^{c,i}, I_l^{c,i}\}_{i=1}^{N_l}, \{L_u^{c,j}, I_u^{c,j}, w_j\}_{j=1}^{N_u}\}$, where $L_l^{c,i}$ and $L_u^{c,j}$ denote the loss values, $I_l^{c,i}$ and $I_u^{c,j}$ denotes the intermediate features for the i-th labeled and j-th unlabeled sample respectively of the c-th class, w_j refers to the weighting score for the j-th unlabeled sample generated from the agent. $N_l : N_u = len(D_l) : len(D_u)$, where len refers to the length of a set. $N_l + N_u = N$, denoting the length of the memory. All memories are constructed in the form of queues, in which new samples are constantly added to replace the oldest samples in each iteration, ensuring that the most recent new samples are always used for computing, catching up with the current training status. Since we only need to compute on the memory with a small length rather than reprocessing all images, the computation cost can be significantly reduced, while maintaining reasonable accuracy with always-new input information close to the current status.

With this memory mechanism, the **average accuracy** of the c-th category used in the state representation can be formulated as the weighted average loss:

$$AL^c = \frac{1}{N_l + \sum_{j=1}^{N_u} w_j} \left(\sum_{i=1}^{N_l} L_l^{c,i} + \sum_{j=1}^{N_u} w_j L_u^{c,j} \right). \tag{4}$$

Thus, the state representation s can be formulated as $\{AL^c\}_{c=1}^{C}$, reflecting the global status of the dataset.

To get the **diversity** used in the action representation, for a sample belonging to the c-th category, we calculate the cosine similarities of its intermediate feature w.r.t all $\{I_l^{c,i}\}_{i=1}^{N_l}$ and $\{I_u^{c,j}\}_{j=1}^{N_u}$ stored in the memory M_c, resulting in N similarity values – the $\{s_l^{c,i}\}_{i=1}^{N_l}$ and $\{s_u^{c,j}\}_{j=1}^{N_u}$. To capture useful statistical information from these similarity values for constructing the action representation, an option is to compute the simple statistics such as the average or maximum value. Here to obtain more informative features, we instead use a histogram to compute the statistical distribution. Specifically, the histogram H has L bins (intervals) with the equal width, so that it has the shape of $\mathbb{R}^L$. Each bin corresponds to a lower bound lb and an upper bound ub. Therefore, each channel H_k of the histogram H can be formulated as:

$$H_k = \frac{1}{N_l + \sum_{j=1}^{N_u} w_j} \left(\sum_{i=1}^{N_l} \mathbb{1}\left(lb_k < s_l^{c,i} \leq ub_k\right) + \sum_{j=1}^{N_u} w_j \mathbb{1}\left(lb_k < s_u^{c,j} \leq ub_k\right) \right), \tag{5}$$

where $k \in [1, L]$, and the bounds lb_k and ub_k can be calculated by:

$$lb_k = \frac{k-1}{L}, \quad ub_k = \frac{k}{L}. \tag{6}$$

Finally, the action representation a for each unlabeled sample can be formulated as $\{\hat{y}^u, EC, H\}$, where $\hat{y}^u$ refers to the pseudo label and EC denotes the entropy confidence. By combining the confidence and diversity information, this representation provides useful reference information implying the potential contribution of each unlabeled sample.

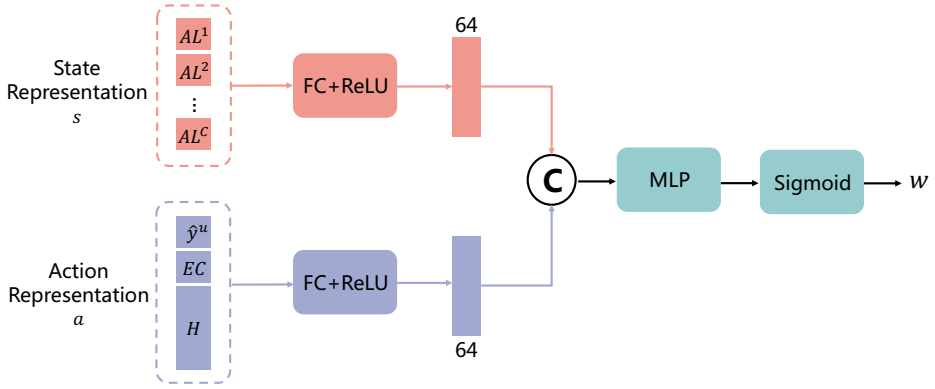


Figure 2: Structure of the agent, where AL^c denotes the average loss value of the c-th class, $\hat{y}^u$ refers to the pseudo label , EC denotes the entropy confidence and H denotes the diversity histogram.

Algorithm 1 Training Phase Algorithm.

Input: an agent network q, a classification network (student network) f, a teacher network g, a dataset $D = \{D_l, D_u, D_r\}$.
while not done **do**
  for $t = 0$ **to** $T - 1$ **do**
    $D_l^t = \{x_i, y_i\}_{i=1}^{N_b}, D_u^t = \{x_i\}_{i=1}^{N_b} \leftarrow$ MinibatchPartition(D_l, D_u)
    Compute supervised loss $\mathcal{L}_l$ from D_l^t using f_t
    Get pseudo labels $Y_u^t = \{\hat{y}_i\}_{i=1}^{N_b}$ from D_u^t using g_t
    Compute state s_t and action a_t using the memory M (Sec. 2.3)
    Generate weighting scores $W_u^t = \{w_i\}_{i=1}^{N_b}$ using q (Sec. 2.4)
    Compute unsupervised loss $\mathcal{L}_u$ from D_u^t, Y_u^t and W_u^t using f_t (Eq. 2)
    Update f_t with the optimization object $\mathcal{L}_u + \mathcal{L}_s$: $f_{t+1} \leftarrow$ UPDATE$(f_t, \mathcal{L}_u + \mathcal{L}_s)$
    Compute reward on D_r: $r_{t+1} \leftarrow$ ACC(f_{t+1}) - ACC(f_t)
    Update memory M
    Update g: $g_{t+1} \leftarrow 0.99 g_t + 0.01 f_{t+1}$
  end for
  Update q following Eq. 3
end while
Return: q

2.4 Agent

With the state and action representations containing the global and sample-specific reference information acquired, the agent q can use them for generating the weighting scores. Specifically, we first feed the state and action representations into two individual linear layers with a ReLU activation, resulting in two features. These two features are concatenated and further processed by a three-layered MLP, whose output passes through a Sigmoid function and becomes the weighting score w. The figure for illustrating the agent structure is presented in Fig. 2.

2.5 Reward

The optimization of the above-introduced agent q can be achieved in a reward-driven manner. Specifically, the reward measures how much the learned policy can benefit the classification network f. It is calculated by evaluating the performance improvement on D_r after one updating step of f from f_t to f_{t+1}, which is formulated as:

$$r_{t+1} = \mathrm{ACC}\left(f_{t+1}\right) - \mathrm{ACC}\left(f_t\right), \tag{7}$$

where ACC denotes the accuracy on D_r. Such a reward directly reflects the performance improvement of f, so that by optimizing using it, the agent can generate a performance improvement-related policy.

2.6 Training and Deployment

With the introduced MDP framework and the designed task-specific state and action representations, training and deploying the agent network are also non-trivial. Typical MDP frameworks Fang et al.

Table 1: Error rates on CIFAR-10, CIFAR-100, SVHN, STL-10 and CIFAR-100-CI datasets. The lower error rate denotes the better performance.

Dataset	CIFAR-10			CIFAR-100			STL-10			SVHN		CIFAR-100-CI
Label Amount	40	250	4000	400	2500	10000	40	250	1000	40	1000	10000
PL Lee et al. (2013)	69.51±4.55	41.02±3.56	13.15±1.84	86.10±1.50	58.00±0.38	36.48±0.13	74.48±1.48	55.63±5.38	31.80±0.29	60.32±2.46	9.56±0.25	45.91±0.32
MixMatch Berthelot et al. (2019b)	47.54±11.50	11.05±0.86	6.42±0.10	67.61±1.32	39.94±0.37	28.31±0.33	-	-	10.41±0.61	42.55±14.53	3.50±0.28	33.09±0.49
ReMixMatch Berthelot et al. (2019a)	19.10±9.64	5.44±0.05	4.72±0.13	44.28±2.06	27.43±0.31	23.03±0.56	37.42±8.44	9.72±1.15	6.64±0.17	3.34±0.20	2.28±0.11	27.01±0.64
AlphaMatch Gong et al. (2021)	8.65±3.38	4.97±0.29	-	38.74±2.94	25.37±0.22	-	-	-	-	3.57±3.13	-	-
UDA Xie et al. (2020)	7.33±2.03	5.11±0.07	4.20±0.12	44.99±2.28	27.59±0.24	22.09±0.19	37.31±3.03	12.07±1.50	6.65±0.25	4.40±2.31	**1.93**±0.01	25.17±0.22
FixMatch Sohn et al. (2020)	6.78±0.50	4.95±0.07	4.09±0.02	46.76±0.79	28.15±0.81	22.47±0.66	35.42±6.43	10.49±1.03	6.20±0.20	4.36±2.16	1.97±0.03	25.20±0.63
Ours	**4.85**±0.09	**4.73**±0.10	**3.79**±0.04	**31.40**±1.99	**23.01**±0.27	**19.18**±0.15	**9.95**±2.69	**7.07**±0.49	**5.17**±0.22	**3.30**±0.98	1.95 ±0.05	**19.65**±0.22

(2017); Padmakumar et al. (2018) consist of a "training stage" where the agent network is pretrained, and a "deployment stage" where the agent network is *fixed* and functioning. However, in our task, such a scheme has to be altered. Although the reweighting policy can be obtained through the initial training stage, the constantly updated classification network prevents the agent from effectively handling the current state and action representations in the deployment phase. Facing the challenge, we design a *pretraining – boosting* curriculum for training and deploying the agent network. In the training stage, as shown in Alg. 1, we pretrain the agent to get an initialized policy. In the deployment stage, instead of using the fixed agent, we update the agent based on the current state and action representations. Specifically, the classification network f and the agent network q are trained alternately. After training f for one epoch, we update q for one step. Such an alternate training mechanism enables to constantly boost the agent, so that the agent can always catch up with the current training status of the classification network, avoiding the policy degeneration. Our experiments validate the effectiveness of such the *pretraining – boosting* curriculum, which demonstrates good performance. To utilize the labeled samples in the reward set D_r, in the last 10% epochs of the deployment phase, we add D_r into the labeled training set D_l, using all labeled images for the semi-supervised classification training.

3 EXPERIMENTS

We evaluate our method on several commonly used semi-supervised learning datasets, including CIFAR-10/100 Krizhevsky et al. (2009), SVHN Netzer et al. (2011), STL-10 Coates et al. (2011) and ImageNet Deng et al. (2009). To verify the effectiveness of our method, we compare it with other advanced methods such as UDA Xie et al. (2020), MixMatch Berthelot et al. (2019b), FixMatch Sohn et al. (2020), Pseudo-Labeling Lee et al. (2013), FlexMatch Zhang et al. (2021a) and AlphaMatch Gong et al. (2021). These methods either do not consider reweighting different samples Lee et al. (2013), use a fixed threshold of confidence for getting cleaner labels Sohn et al. (2020), or design other hand-crafted strategies to address the issues caused by noisy labels Gong et al. (2021) and class bias Zhang et al. (2021a). In contrast, our method brings a brand new perspective.

For a fair comparison, for the semi-supervised training (the boosting phase), we follow the implementation settings of FixMatch Sohn et al. (2020) and FlexMatch Zhang et al. (2021a) to set the common hyper-parameters such as learning rate and weight decay. The details are presented in the appendix. The bins number of histograms used in action equals to 5. For each class, we put 25% of its labeled samples into the reward set D_r. And the memory length equals 100. Note that, our method is **non-sensitive** to these specific hyper-parameters, as proven by the ablation results shown in the appendix.

For the pretraining phase, the number of training steps is 10^{19} (half of the deployment phase), while other settings remain the same as the deployment phase.

3.1 RESULTS

CIFAR-10, CIFAR-100, SVHN, STL-10 and CIFAR-100-CI. We first evaluate the proposed method on CIFAR-10, CIFAR-100, STL-10, and SVHN, and compare it with other advanced methods. For a fair comparison, we follow the setting of Sohn et al. (2020) by using the Wide ResNet-37-2 as the model structure for STL-10 and the Wide ResNet-28-2 for other datasets. The results of other compared methods are the best results reported by the previous papers. Following FixMatch and AlphaMatch, for experiments on CIFAR-100, we add an additional distribution alignment loss into our proposed method.

Table 3: Ablation experimental results of state and action representations.

Method	CIFAR-100	CIFAR-100-CI
w/ complete state and action	31.40	19.65
State w/o average accuracy	36.11	23.92
Action w/o confidence	36.38	24.12
Action w/o diversity histogram	34.66	23.57
Action w/o diversity histogram w/ diversity statistics	32.95	21.12

We present the comparison results in Table. 1. Results on different partition protocols are presented, where the number of labeled images differs. Our method achieves state-of-the-art performance under most settings, and the performance gain is especially significant under some challenging conditions. For example, for the 400 labeled images setting on CIFAR-100, our method outperforms FixMatch by about 15%. This demonstrates the effectiveness of the proposed method when the labeled images are quite limited.

As previous works do, all the above comparison experiments are based on the settings where all categories have the same number of labeled images. We further test the class-imbalanced case, which is a more challenging and closer to real-world condition for semi-supervised learning. Specifically, we use CIFAR-100 to construct a class-imbalanced labeled dataset named CIFAR-100-CI, with different categories having different numbers of labeled images, ranging from 25 to 200. We evaluate different methods on the constructed CIFAR-100-CI except for the AlphaMatch because the code of it is not released. The results are also shown in Table. 1. We can observe that due to the class imbalance, the performance of previous works significantly worsen, while the accuracy drop is relatively small for our proposed method. This is due to the designed state and action representations for computing reweighting scores, which contain the class-bias information that helps the agent to automatically learn a class-unbiased policy.

ImageNet. We further conduct experiments on ImageNet to verify the effectiveness of the proposed method on large and more complex dataset. Following the setting of Zhang et al. (2021a), we randomly choose 100K samples as the labeled data, with each category having 100 samples. The implementation details also follow Sohn et al. (2020); Zhang et al. (2021a). The length of each memory is 250. We report results after training for 2^{20} iterations in Table. 2. The performance of our method outperforms not only the classical FixMatch, but also FlexMatch using the hand-crafted adjustment strategies.

Table 2: Error rates comparison on ImageNet.

Method	Top-1	Top-5
FixMatch Sohn et al. (2020)	43.08	19.55
FlexMatch Zhang et al. (2021a)	35.21	13.96
Ours	**34.09**	**13.11**

3.2 Ablation Study

In this part, we conduct experiments to verify the effectiveness of different settings and components of the proposed method. Due to the paper length limitation, more ablation study results are presented in the appendix, including the ablation for memory length and reward set length. The weight visualization results for the qualitative analysis are also shown in the appendix.

Ablation of State and Action. State and action are treated as the reference information of the agent to produce the weighting scores. We perform experiments to verify the contribution of each designed component in these two representations. The experimental results are shown in Table. 3. We conduct experiments on CIFAR-100 with 400 labeled samples and CIFAR-100-CI with 10000 labeled samples. Under these two settings, the final models by using the complete state and action representations achieve 31.40 and 19.65 error rates respectively. Removing average accuracy in the state representation increases the error rates to 36.11 and 23.92. The action representation contains two parts: the entropy confidence and the similarities histogram. Removing the entropy confidence increases the error rates to 36.38 and 24.12. And removing the similarities histogram increases the error rates to 34.66 and 23.57. For getting the statistical information from the similarities to represent the class-wise diversity, we employ the histogram rather than the simple statistics. We evaluate the setting of using statistics by concatenating the minimum, maximum, average and variance values to represent diversity. This setting increases the error rates to 32.95 and 21.12, showing the advantages of constructing the histograms.

Ablation of the Pretraining-boosting Mechanism. In this paper we propose a dual-phase MDP framework. In the training phase an agent is pretrained to get an initialized policy, which is continuously optimized in the deployment phase to catch up with the current status. We conduct experiments on CIFAR-100 with 400 labeled samples and CIFAR-100-CI with 10000 labeled samples, to demonstrate the effectiveness of this design. The results are presented in Table. 4. Specifically, we remove the boosting operation but use a fixed agent in the deployment phase. This setting increases the error rates from 31.40 and 19.65 to 33.65 (+2.25) and 22.71 (+3.06). This result verifies that the agent boosting plays an important role in improving the model performance.

Table 4: Ablation experimental results of the pretraining-boosting mechanism.

Method	CIFAR-100	CIFAR-100-CI
w/ pretraining and boosting	31.40	19.65
w/o boosting	33.65	22.71

4 Related Work

In recent years, semi-supervised learning has attracted much attention, since it provides a mechanism to improve model performance using both the labeled and unlabeled data. The traditional semi-supervised learning methods were usually based on the entropy minimization Grandvalet et al. (2005); Lee et al. (2006), transductive models Demiriz et al. (1999); Joachims et al. (1999) and graph-based models Belkin et al. (2004); Wang & Zhang (2007); Zhou et al. (2003); Blum & Chawla (2001).

Most of recent approaches employed the pseudo label training mechanism Lee et al. (2013), where hard pseudo labels obtained from the prediction results are used for supervision. However, this simple mechanism cannot bring satisfactory performance, mainly due to the noisy labels issue caused by the inaccurate prediction results. This issue motivated recent works Rosenberg et al. (2005); Xie et al. (2020); Sohn et al. (2020); Gong et al. (2021) to introduce thresholding strategies to filter out and only use pseudo labels with sufficient confidence. Such thresholding strategies can be treated as the sample reweighting methods Jiang et al. (2018; 2020); Zhang et al. (2020); Ren et al. (2018), through either the soft weighting functions with a temperature Xie et al. (2020); Gong et al. (2021), or the hard weighting one-hot strategy based on the fixed confidence Sohn et al. (2020). Another key issue for semi-supervised learning is caused by the class imbalance, which is especially serious when the labeled samples number is small. To address this issue, FlexMatch Zhang et al. (2021a) set different thresholds for different classes, encouraging the optimizer to pay more attention on the hard categories. The similar motivation was adopted and implemented by Hu et al. (2021), where training data within different classes were randomly sampled with different sample rates computed according to the class's learning status. These methods are often based on the heuristic rules, relying on hand-crafted thresholding or sample reweighting schemes. In addition to the basic classification task, such the hand-crafted schemes were also used in semi-supervised learning for other downstream tasks such as object detection Liu et al. (2021); Zhang et al. (2021b), semantic segmentationHu et al. (2021); Yang et al. (2021); He et al. (2021) and action recognition Singh et al. (2021); Xiao et al. (2021).

In this paper, motivated by the recent success of applying reinforcement learning on different tasks Fang et al. (2017); Padmakumar et al. (2018); Bachman et al. (2017); Le et al. (2021); Arulkumaran et al. (2017), we free the need for hand-crafted strategies designing as the above-mentioned methods, and propose a brand new method for semi-supervised learning by using a Markov Decision Process framework solved by reinforcement learning.

5 Conclusion

In this paper, we develop a novel semi-supervised learning method from a brand new perspective, where an end-to-end learned sample reweighting policy is employed, enabled by a novel MDP framework. We delicately design the state-action representations and the "pretraining-boosting" dual phase training strategy. Also, the queued memories are introduced to relieve the computation burden. Extensive experiments show that our method achieves outstanding performance, outperforming the previous advanced approaches on multiple datasets.

REFERENCES

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

Philip Bachman, Alessandro Sordoni, and Adam Trischler. Learning algorithms for active learning. In *international conference on machine learning*, pp. 301–310. PMLR, 2017.

Mikhail Belkin, Irina Matveeva, and Partha Niyogi. Regularization and semi-supervised learning on large graphs. In *International Conference on Computational Learning Theory*, pp. 624–638. Springer, 2004.

David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution matching and augmentation anchoring. In *ICLR*, 2019a.

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. *NeurIPS*, pp. 5050–5060, 2019b.

Avrim Blum and Shuchi Chawla. Learning from labeled and unlabeled data using graph mincuts. 2001.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, pp. 215–223, 2011.

Ayhan Demiriz, Kristin P Bennett, and Mark J Embrechts. Semi-supervised clustering using genetic algorithms. *Artificial neural networks in engineering (ANNIE-99)*, pp. 809–814, 1999.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pp. 248–255. Ieee, 2009.

Meng Fang, Yuan Li, and Trevor Cohn. Learning how to active learn: A deep reinforcement learning approach. *arXiv preprint arXiv:1708.02383*, 2017.

Chengyue Gong, Dilin Wang, and Qiang Liu. Alphamatch: Improving consistency for semi-supervised learning with alpha-divergence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13683–13692, 2021.

Yves Grandvalet, Yoshua Bengio, et al. Semi-supervised learning by entropy minimization. In *CAP*, pp. 281–296, 2005.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Ruifei He, Jihan Yang, and Xiaojuan Qi. Re-distributing biased pseudo labels for semi-supervised semantic segmentation: A baseline investigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6930–6940, 2021.

Hanzhe Hu, Fangyun Wei, Han Hu, Qiwei Ye, Jinshi Cui, and Liwei Wang. Semi-supervised semantic segmentation via adaptive equalization learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning*, pp. 2304–2313. PMLR, 2018.

Lu Jiang, Di Huang, Mason Liu, and Weilong Yang. Beyond synthetic noise: Deep learning on controlled noisy labels. In *International Conference on Machine Learning*, pp. 4804–4815. PMLR, 2020.

Thorsten Joachims et al. Transductive inference for text classification using support vector machines. In *Icml*, volume 99, pp. 200–209, 1999.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Ngan Le, Vidhiwar Singh Rathour, Kashu Yamazaki, Khoa Luu, and Marios Savvides. Deep reinforcement learning in computer vision: a comprehensive survey. *Artificial Intelligence Review*, pp. 1–87, 2021.

Chi-Hoon Lee, Shaojun Wang, Feng Jiao, Dale Schuurmans, and Russell Greiner. Learning to model spatial dependency: Semi-supervised discriminative random fields. *Advances in Neural Information Processing Systems*, 19, 2006.

Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, 2013.

Yen-Cheng Liu, Chih-Yao Ma, Zijian He, Chia-Wen Kuo, Kan Chen, Peizhao Zhang, Bichen Wu, Zsolt Kira, and Peter Vajda. Unbiased teacher for semi-supervised object detection. *arXiv preprint arXiv:2102.09480*, 2021.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

Aishwarya Padmakumar, Peter Stone, and Raymond J Mooney. Learning a policy for opportunistic active learning. In *EMNLP*, 2018.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International conference on machine learning*, pp. 4334–4343. PMLR, 2018.

Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. Semi-supervised self-training of object detection models. 2005.

Ankit Singh, Omprakash Chakraborty, Ashutosh Varshney, Rameswar Panda, Rogerio Feris, Kate Saenko, and Abir Das. Semi-supervised action recognition with temporal contrastive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10389–10399, 2021.

Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *NeurIPS*, 33, 2020.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Fei Wang and Changshui Zhang. Label propagation through linear neighborhoods. *IEEE Transactions on Knowledge and Data Engineering*, 20(1):55–67, 2007.

Junfei Xiao, Longlong Jing, Lin Zhang, Ju He, Qi She, Zongwei Zhou, Alan Yuille, and Yingwei Li. Learning from temporal gradient for semi-supervised action recognition. *arXiv preprint arXiv:2111.13241*, 2021.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. Unsupervised data augmentation for consistency training. *NeurIPS*, 33, 2020.

Lihe Yang, Wei Zhuo, Lei Qi, Yinghuan Shi, and Yang Gao. St++: Make self-training work better for semi-supervised semantic segmentation. *arXiv preprint arXiv:2106.05095*, 2021.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.

Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. *Advances in Neural Information Processing Systems*, 34, 2021a.

Fangyuan Zhang, Tianxiang Pan, and Bin Wang. Semi-supervised object detection with adaptive class-rebalancing self-training. *arXiv preprint arXiv:2107.05031*, 2021b.

Zizhao Zhang, Han Zhang, Sercan O Arik, Honglak Lee, and Tomas Pfister. Distilling effective supervision from severe label noise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9294–9303, 2020.

Dengyong Zhou, Olivier Bousquet, Thomas Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. *Advances in neural information processing systems*, 16, 2003.

A Appendix

A.1 Qualitative Analysis

To obtain more insights into how our method works, we further visualize the relationship between the weight decision with the state and action representations in different training epochs. The results are shown in Fig.3, where orange dots denote the high-weight samples/classes and the blue dots denote other samples/classes with smaller weights. In our work, state representation is measured by the average accuracy and action representation is composed of the diversity and confidence.

As can be observed from (a) and (b), dots closer to the center denote samples with the lower diversity. As shown by the orange dots, in the beginning stage of training, most orange dots are located in the center region, reflecting that the learned policy tends to give the larger weighting scores to the lower-diversity samples (i.e., the more common samples). Such samples are more common in the dataset, helping the model to quickly fit to an acceptable performance. In the ending stage of training, the policy gives higher weighting scores to samples with the higher diversity, since the optimizer needs to pay more attention on such samples for avoiding overfitting and getting a more generalized model.

As can be observed from (c) and (d), dots closer to the center denote samples with the lower confidence. in the beginning stage, the dots are evenly distributed in the visualization space, indicating the existence of samples with different confidence. In this stage, most orange dots are distributed in the area far away from the center, implying that the policy prefers to give higher scores to the high-confidence samples. In the ending stage, the model becomes more reliable so that the confidence of all samples are relatively large. In this case, most orange dots are also distributed in the area far away from the center. Therefore, it can be inferred that the learned policy always tend to select the high-confidence samples for the more reliable semi-supervised learning.

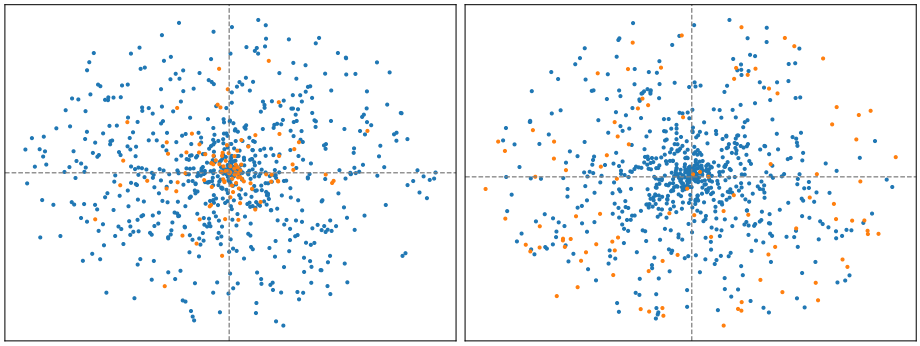
As can be observed from (e) and (f), dots closer to the center denote classes with the lower average accuracy. In both the beginning and ending stages, most orange dots that indicate classes with the highest average weights are distributed in the area close to the center. This shows that the policy tends to give higher weight scores to classes with lower average accuracy, helping the network to improve such classes with bad performance.

A.2 Implementation Details

For reproduction, as shown in Table. 5, we present the hyperparameter settings of our proposed method. Note that most settings follow the Zhang et al. (2021a) and Sohn et al. (2020).

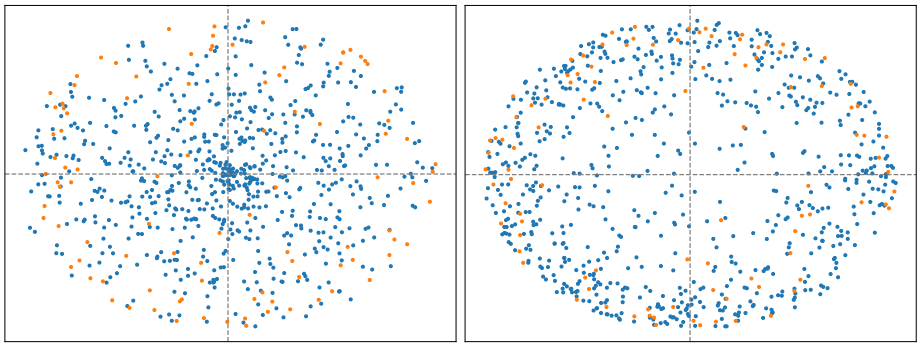
Table 5: Hyperparameter settings for different datasets. Most settings follow the Zhang et al. (2021a) and Sohn et al. (2020).

Dataset	CIFAR-10	CIFAR-100	STL-10	SVHN	CIFAR-100-CI	ImageNet
Model	WRN-28-2 (Zagoruyko & Komodakis, 2016)	WRN-28-2	WRN-37-2	WRN-28-2	WRN-28-2	ResNet-50 (He et al., 2016)
Weight Decay	5e-4	1e-3	5e-4	5e-4	5e-4	1e-3
Batch Size	64					128
Learning Rate	0.03					
SGD Momentum	0.9					
EMA Momentum	0.99					
Number of Diversity Histogram Bins	5					
Reward Set Length	1/4 of the labeled image number					
Pretraining (training) Phase Iteration Number	2^{21}					
Boosting (deployment) Phase Iteration Number	2^{20}					
γ in TD Loss	0.9					



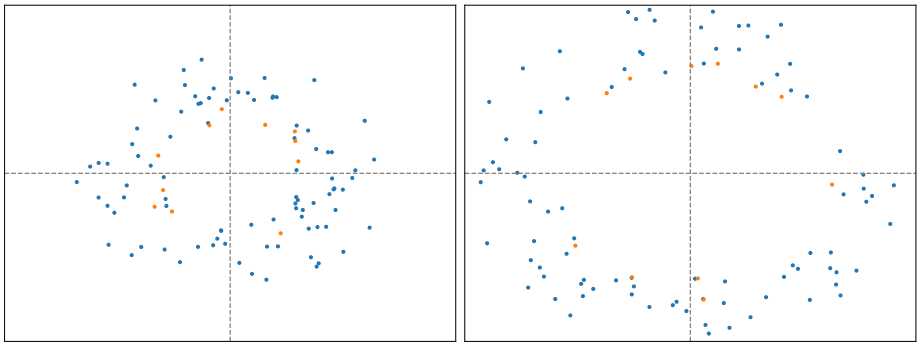
(a) Qualitative result of **diversity** at the beginning stage

(b) Qualitative result of **diversity** at the ending stage



(c) Qualitative result of **confidence** at the beginning stage

(d) Qualitative result of **confidence** at the ending stage



(e) Qualitative result of **class average accuracy** at the beginning stage

(f) Qualitative result of **class average accuracy** at the ending stage

Figure 3: Best viewed in color. Qualitative analysis of how different components in the state and action representations affect the weighting scores. The left and right columns present the results at the beginning and ending stages, respectively. The orange dots denote the samples/classes with the largest (average) weighting scores. Dots closer to the center denote samples with the lower diversity, the lower confidence and classes with the lower average accuracy, respectively.

B More Details of the Temporal Difference (TD) Error

We use the temporal difference (TD) error to optimize the agent network q. In the text, due to the paper length limitation, for simplicity, loss function is formulated under the setting where only sample participates in the calculation:

$$TD\left(\theta, \hat{\theta}\right) = \left(r_{t+1} + \gamma q\left(s_{t+1}, a_{t+1}; \hat{\theta}\right) - q\left(s_t, a_t; \theta\right)\right)^2. \tag{8}$$

Actually in practice, we use the mini batch strategy with N_b unlabeled samples being processed simultaneously. Therefore in this case, the loss function can be reformulated as:

$$TD\left(\theta,\hat{\theta}\right)=\left(r_{t+1}+\gamma\max_{\{a_{t+1}^{j}\}_{j=1}^{N_b}}q\left(s_{t+1},a_{t+1}^{j};\hat{\theta}\right)-\max_{\{a_{t}^{j}\}_{j=1}^{N_b}}q\left(s_{t},a_{t}^{j};\theta\right)\right)^{2}, \tag{9}$$

where a_t^j denotes the action representation of the j-th unlabeled sample in a mini batch.

We do not optimize the agent after processing each mini batch, which consumes too much computation. Instead, as shown in Alg. 1 in the text, we update the agent after training the classification network for one epoch. We follow Van Hasselt et al. (2016) by employing the experience replay mechanism for training. Specifically, considering an epoch with T iterations, in the t-th iteration we get $\mathcal{T}_t=\{s_t,\{a_t^j\}_{j=1}^{N_b},r_{t+1},s_{t+1}\}$. So in the whole epoch, we can get an experience replay buffer ε storing $\{\mathcal{T}_t\}_{t=1}^{T}$. In the end of each epoch, we randomly sample several $\mathcal{T}_t$ from ε for computing the loss. Thus, the final TD loss can be reformulated as:

$$TD\left(\theta,\hat{\theta}\right)=\mathbb{E}_{\mathcal{T}_t\sim\varepsilon}\left(r_{t+1}+\gamma\max_{\{a_{t+1}^{j}\}_{j=1}^{N_b}}q\left(s_{t+1},a_{t+1}^{j};\hat{\theta}\right)-\max_{\{a_{t}^{j}\}_{j=1}^{N_b}}q\left(s_{t},a_{t}^{j};\theta\right)\right)^{2}. \tag{10}$$

In practice, γ equals 0.9 and the number of sampled $\mathcal{T}_t$ for computing loss equals 8.

B.1 Sensitivity Analysis

Our method requires some hyper-parameters including the histogram bin number, the reward set length and the memory length. Here we conduct experiments to verify that our method is **non-sensitive** to such hyper-parameters. Experiments are performed on CIFAR-100 with 2500 labeled images.

B.1.1 Histogram Bin Number

As can be observed from Table. 6, the performance is insensitive to the bin number. When the number is larger than 3, our method keeps stable and can consistently achieve good performance. It is worth noting that on all the tested settings, the performance of our method can significantly outperform that of the baseline method FixMatch (with 28.15 error rate), showing the robustness of our method.

Bin Number	2	3	4	5	6	8	10
Error Rate	24.17	23.55	23.12	**23.01**	23.14	23.09	23.20

Table 6: Ablation results of histogram bin number.

B.1.2 Reward Set Length

In our method, the reward set length equals 25% of the labeled images number. We conduct experiments to verify different length settings and the results are shown in Table. 7. The results indicate that our method is non-sensitive to this parameter, as the performance fluctuates less when the length ranges from 10% to 40% of the labeled images number.

Reward Set Length	10%	15%	20%	25%	30%	35%	40%
Error Rate	23.85	23.77	23.37	**23.01**	23.20	23.19	23.56

Table 7: Ablation results of the reward set length.

B.1.3 Memory Length

The memory length in our method is set to 100. From the ablation results as presented in Table. 8, we observe that once the length is larger than 50, the performance can keep stable, and further increasing the length cannot achieve significant improvement.

Bin Number	40	50	75	100	150	200	300
Error Rate	23.50	23.17	23.12	**23.01**	23.24	23.05	23.10

Table 8: Ablation results of histogram bin number.

B.2 Hand-crafted Strategies Exploiting the Same Information as the Policy Agent

One crucial contribution of our method is to automatically learn a rweighting policy using the RL mechanism. To evaluate the effect of the automatically learning mechanism, here we evaluate the method by using the same information (average accuracy A, confidence C and diversity D) to design the hand-crafted strategies. Experiments are conducted on CIFAR-100 with 25 labeled images per class. Specifically, by normalizing and adding all these indicators, a score s is generated, formulated as $s = \alpha A + \beta C + \gamma D$, where α, β and γ are three hyper-parameters. Considering all these information may be positively or negatively correlated with the weight, we test different groups of settings, in which α, β and γ range from -1 to 1, where the negative and positive values represent the negative and positive correlations between the information and the generated weights, respectively. We also introduce a manually-set proportion p, such that only the p unlabeled samples with the highest s are selected for training. We test different p ranging from 20% to 80%. In all the tested settings, the error rates range from 26.09 to 33.72. The lowest error among all tested settings of these hand-crafted strategies is still significantly higher than our proposed method (23.01).

As discussed above in the qualitative analysis, the model tends to give the higher scores to the low-diversity samples in the beginning stage and high-diversity samples in the ending stage. Based on such conclusions, we further design a hand-crafted strategy where γ is -1 in the first 50% epochs and 1 in the last 50% epochs. Such a method achieves 25.30 error rate, which is better than all other hand-crafted strategies, but still significantly worse than our proposed method.

The above results demonstrate that, it is really hard to manually design a hand-crafted strategy that makes use of the same information used by the policy agent and can achieve good performance, further showing the advantage of our MDP-based method that dose not need to design hand-crafted strategies.

B.3 Training Time

Our method takes about 1.56 times as long as the baseline method FixMatch. Though needing more but acceptable extra training time, it can outperform FixMatch significantly on different datasets, as shown in Table 1 of our paper. Also note that, after the semi-supervised training is finished, at the inference stage, the time for each prediction operation is the same as FixMatch.